Expert Guided Adaptive Maintenance

Tony Lindgren[1] and Jonas Biteus[2]

[1]*Department of Computer and System Sciences, Stockholm University, Forum 100, 164 40 Kista, Sweden*
tony@dsv.su.se

[2]*Scania CV AB, Service Support Solutions, YSPX, Verkstadsvägen 17, by 280, 151 87 Södertälje, Sweden*
jonas.biteus@scania.com

ABSTRACT

The heavy truck industry is a highly competitive business field; traditionally maintenance plans for heavy trucks are static and not subject to change. The advent of affordable telematics solutions has created a new venue for services that use information from the truck in operation. Such services could for example aim at improving the maintenance offer by taking into account information of how a truck has been utilized to dynamically adjust maintenance to align with the truck's actual need. These types of services for maintenance are often referred to as condition based maintenance (CBM) and more recently Integrated Vehicle Health Management (IVHM).

In this paper we explain how we at Scania developed an expert system for adapting the maintenance intervals dependent on operational data from trucks. The expert system is aimed at handling components which maintenance experts have knowledge about but do not find it worth the effort to create a correct physical wear-model for.

We developed a systematic way for maintenance experts to express how operational data should influence the maintenance intervals. The rules in the expert system therefore are limited in what they can express, and as such our presented system differs from other expert systems in general.

In a comparison between our expert system and another general expert system framework, the expert system we constructed outperforms the general expert framework using our limited type of rules.

1. INTRODUCTION

Expert systems have been around for a long time (Durking, 1990; Russel & Norvig, 2010). They have been successfully used in a variety of applications ranging from diagnosing medical problems (Buchanan & Shortliffe, 1984) to facilitate space exploration (Marsh, 1988). Today the term expert system is not used to any large extent, especially not in industry, now days they are often referred to as rule engines. In this paper we will use the term expert system and not rule engine.

Scania Commercial Vehicles (Scania) is a manufacturer of heavy trucks, coaches and engines for industrial and marine usage. We at Scania have investigated how an expert system could be used for improving the maintenance of our products. The aim is to achieve perfect alignment with the maintenance program of a Scania product with the actual maintenance needs of the product. Using on-board sensors from our vehicles we collect data of how the vehicle is utilized. This operational data together with expert knowledge, captured in a type of expert system, is used to adapt the maintenance program to match each vehicle individual maintenance needs.

This paper is focused on describing the design considerations developing such adaptive system. We also relate our system to other general expert systems. The rest of the paper is organized as follows: In the next section we will look at rules and expert systems in greater detail. Then the current solution of vehicle maintenance at Scania is presented after that we present our proposed solution, its implementation and the findings from comparing it with a general expert system. The last section is dedicated to discussion and conclusions and finally we give pointers to future work.

2. EXPERT SYSTEMS AND RULES

An expert system has two major settings of operation, one when the knowledge base is updated and one when using the knowledge base.

In the former case an expert's knowledge of a domain is captured and inserted through the user interface. The information is then stored in the knowledge base in a

Tony Lindgren et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

suitable format for the inference mechanism. In the latter case a user (or computer) post a question using the user interface and the inference mechanism infer an answer which is presented for the user.

Expert systems are beneficial when developing advanced software systems because they fulfill the need of separating out the expert knowledge from the source code. This separation is typically beneficial for easy maintenance of the knowledge base over time. Re-use of proven inference mechanisms is also facilitated using this approach as the inference mechanism can be an external software module.

2.1. Rules

Rules come in different flavors, but there are two dominant types, production rules and logic programming rules. As noted in the paper by Kowalski and Sadri (Kowalski & Sadri, 2009), these two types of rules have traits which overlap but also have differences between them. In this paper we will make a simple distinction between them and use the term production rules for rules which use a forward-chaining inference mechanism and logic programming rules as rules which use a backward-chaining inference mechanism. For a clarifying paper about these inference mechanisms, see (Shapiro, 1987).

Basically the main difference between the two inference mechanisms is how search is conducted. In a search problem setting we have a certain goal and a current state, i.e. where we are now. If we choose to search from the current state until we find the goal, we are doing forward-chaining inference. If we start from the goal and search (backward) until we find a path to the current state, we are doing backward-chaining inference.

In a rule based system this type of search are conducted in a knowledge base together with a question or new fact. The type of inference mechanism is closely related to what type of reasoning we are interested in. For example are we interested in answer(s) to a certain question or do we want to see the implications of new facts that we just observed?

Typical heuristics for choosing one inference mechanism is to consider what event that trigger the problem solving. If the trigger is a new fact then the exploration of consequences given the new fact is naturally handled by forward-chaining mechanism. If on the other hand the trigger is a query to which an answer is required they are naturally handled by backward-chaining.

Other general rules for guidelines for choosing inference mechanism are to investigate the branching of the search space, i.e. depending upon the knowledge base. If the average state in the search space has more successors than predecessors backwards-chaining is desirable. If the average state has fewer successors than predecessor it is desirable to use forward-chaining. These two inference mechanisms can also be mixed.

3. HEAVY TRUCK MAINTENANCE – CURRENT SITUATION AT SCANIA

Today the maintenance plan for Scania vehicles is set when the vehicle is sold. This is typically done by sales personnel together with the buyer by selecting one of a set of predefined maintenance plans that best matches the vehicle specifications and the buyers intended usage.

The predefined maintenance plans are developed and maintained by skilled personnel having knowledge about both the products and customer's usage. Vehicle usage is divided into six typical applications types. For each application type and vehicle specification, a cyclic maintenance plan is given as the number of kilometers between maintenance occasions with fixed maintenance protocols.

Maintenance is always done in a cycle of S-M-S-L occasions, where S = Small, M = Medium, and L = Large are different maintenance modules for maintaining different sets of components.

There are a number of problems with the way maintenance plans are created today:

1. Much responsibility is put on the sales personnel to know the product as well as the customer's usage of the product.
2. Once created the plans are seldom updated even if the application of the vehicle changes. Thus, it is possible that the maintenance a vehicle receives does not correspond to its needs.
3. Although the fixed S, M, and L modules make it convenient to plan, they contain maintenance points that do not need to be grouped together with the effect that some components are maintained more than necessary.
4. The current maintenance plans are coarse in the sense that the precision in the type of application must be fitted into one of the six types of application. Therefore the experts dictating when maintenance ought to be done, use a safety margins given the uncertainty of the actual usage of a particular vehicle. This has two consequences, one is that plans are not individualized to the degree that they could be and the second consequence is that components are maintained more than necessary.

4. PROPOSED SOLUTION'S SCOPE, AIMS AND MOTIVATION

Many problems with the current situation can be improved with a system for Integrated Vehicle Health Monitoring (IVHM), see (Ian K. Jennions et al., 2011; Dunsdon & Harrington, 2008). Using modern IT technologies communication between Scania trucks and our system is feasible. This includes acquiring operational data from a specific trucks while in operation, this data can then be used to calculate the maintenance need of a vehicle.

This computation of the maintenance need can be done in a verity of ways with different complexities. Ranging from computer models that capture the maintenance point physical characteristics to simple preset deadlines, dependent on some operational data, which dictate when maintenance should be done.

The aim of our expert system was to capture the knowledge of our maintenance experts in a systematic and user friendly manner. The system was design so that the maintenance experts should be able to edit "rules" them self and also able to verify them.

The intension of the system was that it should be used when experts "know" how operational factors, measured via operational data, affect the maintenance need of a component, but we are not interested in creating a complex and fully verified maintenance model for the component. The reasons of why we want to use the expert system and don't want to create a "full" model can be motivated by the fact that the cost of creating such a model is regarded as to high compared to its benefits.

As a truck from Scania consist of around 80 to 160 unique maintenance points related to different components. Currently we have created four "full" maintenance models for components with vital importance and this figure will probably rise in the future. But for maintenance points that will not have "full" models an expert system seems like a logical way to address the need for individualized maintenance from a technical and business oriented view.

4.1. Expert system design

To create our expert system we firstly removed the maintenance points from their S, M and L modules and let the maintenance experts themselves define new maintenance points. Thus improving the precision as maintenance point no longer needs to be lumped together.

When experts express rules regarding maintenance points they need to convey information about "*which specification is the maintenance point valid for?"* and *"when is the maintenance point valid?"* The first question is specified by part-numbers used by Scania when assembling a truck. The second question is specified by intervals utilize three basic types of information; *mileage*, *operational hours* and *static time.*

Mileage is self-explanatory, operational hours is defined as time when the engine is running and static time denotes calendar time. For example can an interval be defined by **opHours_cond(0,+inf)**, which denotes that a rule is valid for a whole vehicles lifetime, as it is valid from 0 operating hours to infinity (inf) operating hours. Mileage is measured in km, operational time in hours and static time in days.

This interval validity condition was requested by the maintenance experts as they wanted to be able to express different rules for different ages of a component, i.e. check a chassis for cracks do not to happen frequently when a truck is new but when it's old it needs to be done more often. This type of rule also put a demands on the system to keep track of events which causes reset of the three type of conditions, for example even how unlikely it may be, if we replaced the chassis on a truck.

4.1.1. Operational data

Before we look in to how the experts can use operational data to influence the maintenance point we have to look at the characteristics of operational data at Scania trucks.

Operational data is captured in episodes, i.e. from time t_0 to time t_1. These episodes can be of varying length, trucks with wireless telemetric can export episodes at a preset periodicity, while trucks not having wireless telemetric might have episodes that are equal to their operational time between workshop visits.

Three different data formats exist for operational data, scalar, vector and matrix format. Measurement for average fuel consumption is a scalar value, i.e. **fuel_consumption(53)** = 53 liter / 100 km, which is calculated for an episode. Vectors can for example be the **altitude(VeryLow, Low, Medium, High, VeryHigh)**, VeryLow = less than 110m, Low = 110m to 990m, Medium = 990m to 1950m, High = 1950m to 3000m, VeryHigh = 3000m or more over sea level.

The value for operational data variables are aggregated in bins and reflect the amount of time the truck is used under conditions for a particular bin. The same is true for matrix bins, but each bin has two conditions to adhere to. For example we could have a matrix measuring load in tons on the y-axis and speed of the truck on the x-axis.

4.1.2. Expressing how operational data influence maintenance

Using the three types of operational data collected from Scania products maintenance experts can via rules express

boundaries for deadlines and how operational data should influence maintenance deadlines.

Rules have boundaries to make the maintenance point rules well defined, each rule has a minimum, maximum and a base value for at least one of three basic types and all three basic types can have these values. These values define the deadline span of the maintenance point and its central value, i.e. the base value. The application of the rule can never result in a lower value than the minimum value defined or higher than the maximum value defined.

For example can we use the basic type km and define the minimum value = 50 000 km, base value = 100 000 km and maximum value = 150 000 km.

Using numbers in range [-9, 9], the users can express how one instance of operational data influence the basic types for a specific rule. For example if we have the expression **altitude(4, 2, 1, -3, -8)** with the base values as defined above and an operational data episode from a truck with the following values **altitude(0.1, 0.3, 0.6, 0, 0)**, where the aggregated values are normalized. This outcome for a rule is calculated in two steps, first the impact score in this case 4 * 0.1 + 2 * 0.3 + 1 * 0.6 + -3 * 0 + -8 * 0 = 1.6 then the we apply the impact score onto the basic types, in this case as the value is positive 150000 – 100000 / 9 = 5556, 5556 *1.6 + 100000 = 108889. Hence in this case the system would output 108889 km as deadline for this particular maintenance point.

More generally the impact is calculated as follows:

$$\frac{\sum_i^{op\_factors} \sum_j op\_factors_{ij} \times op\_values_j}{|op_{factors}|} \quad (1)$$

Where the op_factors is the operational data influence specified by the maintenance expert, ranging from -9 to 9. The value can be set when answering the question *"how much impact should we assign to observing this operational data in the relation to the base value and in what direction?"*

Calculating the basic type outcome given impact:

$$imp\_v \begin{cases} \geq 0, \frac{max\_v - base\_v}{9} \times imp\_v + base\_v \\ < 0, base\_v - \frac{base\_v - min\_v}{9} \times imp\_v \end{cases} \quad (2)$$

The maintenance expert is free to set the basic types base value anywhere between the minimum value and the maximum value. If it is set in the middle of these two values the "steps" will be equally long on each side of the base value, i.e. an impact value of 2 and -2 will amount in the same increase respectively decrease in the basic type. Setting the base value allows the expert to change how the impact will affect the outcome.

When the impact is zero the outcome is the base value and when it is 9 the outcome is the maximum value and -9 correspond to the minimum value. When experts define rules and use vector and matrix data which are distributions, it is unlikely that the impact will come close the endpoints of (+/-) 9.

However scalars do not have any predefined bins and it is up to the experts to create the bins and set the influence value of (-/+) 9 for each bin. For example **fuel_consumprion(from, to, influence_value)**, where *from* and *to* define the lower resp. higher bound for the bin. The scalars behave differently from vectors and matrix distributions in that one bin will get a 1 and the rest of the bins zero. Hence the influence value should be set with caution for scalars.

In conclusion a maintenance expert defines the following values for a rule: **ValidSpecification, BasicRuleIntervalCondition, Min, Base, Max, ExpertMaintInfluenceList.**

5. IMPLEMENTATION

We implemented the system in SICStus Prolog, see (Mats Carlsson et al., 2013). One of the motivations of choosing this language is that Prolog uses a backwards chaining proof (resolution) to prove questions posted to it together with goals and facts in its knowledge base (or program). This fits fine with our intention of creating a system that answers the maintenance needs given a trucks operational data and specification.

Using this programming language you get a complete and sound and tested theorem-prover "for free", which made it an ideal language for our purposes. Other expert system frameworks could have been chosen, which we will elaborate further upon at the end of the paper, but the primary reason is our limited and restrictive "rules", that did not need any fancier expert framework.

To ensure better modularity we used the Rule Interchange Format (RIF) (W3C, 2013) standard proposed by W3C. The standard is supported by a number of expert systems, for example IBM Websphere and ILOG JRules, OntoBroker, Oracle Business Rules (OBR) etc. To ensure backwards compatibility and development of new knowledgebase releases, we utilized Prologs blackboard functionality, using **version** and **status** as keys to a certain blackboard. Version is just a version number and status can be one of

development, *testing* and *released*. Essentially a blackboard is a memory area where we post one knowledge base.

We created a webb-based GUI using AJAX technology for creating and simulating rules. The GUI also check rule validity, i.e. that the base value is higher than the minimum value and lower than the maximum value.

The expert system is a separate module and runs on a server exposing its services through the PrologBeans interface. Our solution make is possible to keep track of different user sessions and service many requests simultaneously. We have aimed for the modules to be self-contained with a clear interface. The expert system has two main services, loadRules and useRules. Loading a rule set check that it is syntactical correct, while semantics are pushed to the GUI, i.e. checks that intervals are defined correct etc. We also facilitate expert's creation of new rule sets and updates to existing rule sets by addRule and removeRule.

6. COMPARISON WITH OTHER SYSTEMS

The initial motivation for using a programming language as Prolog for implementing the expert system was to have the freedom to change the system depending on the need from the users and to explore different solutions to the problem.

There are a number of different expert systems available, both commercial and open-source. There does not exist, to the author's knowledge at least, a multitude of systematical comparisons of expert systems. But one comparison of rule engines has been done in the field of Semantic Web which recommended for the interested reader (Senlin Liang, et al., 2009).

One of the more successful open-source tools is Drools (Red Hat, 2013). Drools is part of the JBoss platform. It is an open-source software that aims at being "…a unified and integrated platform for Rules, Workflow and Event Processing". To investigating how our expert system performance is comparable to other established general expert systems, we choose Drools to compare with. The reason was mainly its availability as it is open source software and partly because it is well established.

The experimental setup was as follows:
We implemented the same type of reasoning in Drools as we do in our system. Then we created knowledge base's consisted of a base set of 1000 rules, each of these rules hade truck specification conditions (TSC) **not** matching an intended query. Into this knowledge base we injected rules at random that had TSC that matched the intended query. The TSC consisted of: **ValidSpecification** and **BasicRuleIntervalCondition** as mentioned before. Two **ValidSpecification** conditions were used for all rules. The number of the injected rules, where 60, 80 and 100. This procedure was repeated 10 times, so in total 30 rule bases was created, each with an random injection of rules, and equally many queries was made. For each query the CPU time was measured and the amount of memory used.

In Table 1 the amount of time (in milliseconds) for each system to answer a query is shown. The number of matches at each query is 60, 80 and 100 respectively. The minimum, average and maximum time is shown for the 10 queries.

Table 1. The minimum, average and Maximum CPU time in milliseconds used to answer the 10 queries with 60, 80 and 100 matches.

	60			80			100		
	MI	AV	MA	MI	AV	MA	MI	AV	MA
Drools	172	179	204	188	206	298	204	229	313
Prolog	109	129	187	109	125	156	109	139	187

The memory consumption is always the same when using Prolog, probably because its allocated memory in chunks and the different sizes of rule set does not render in need of more memory allocation. Drools on the other hand allocate different memory sizes on each run. See Table 2 for an overview, using the same structure as in Table 1 but measuring the memory needs in megabytes.

Table 2. The minimum, average and maximum memory used in megabytes used to answer the 10 queries with 60, 80 and 100 matches.

	60			80			100		
	MI	AV	MA	MI	AV	MA	MI	AV	MA
Drools	120	197	247	53	203	253	136	205	267
Prolog	3,9	3,9	3,9	3,9	3,9	3,9	3,9	3,9	3,9

One possible reason for the big memory needs for Drools compared with Prolog is that the rules cannot be written as compact as in Prolog. In Prolog a rule is one line, in Drools the same rule is written in around 40 lines. This extra size of the rule set is possibly an explanation of the extra time needed by Drools to answer the queries.

From the experiment it evident that our system outperforms Drools, both when it comes to response times and memory consumption.

7. CONCLUSION

We have presented a systematic way of capturing expert's knowledge in the field of heavy truck maintenance. The suggested way of making use of expert knowledge through an expert system is motivated for the bulk of components

that we want to maintain, but do not want to create an advanced model for.

To achieve adaptive maintenance for vehicle's components we think our solution has a given place when considering a balance of cost and speed of creating rules in our system compared to more advanced models. Thus we believe this approach will be a starting point for adaptive maintenance for a majority of components.

The implementation we made also showed that our solution outperforms a leading off the shelf product. This is encouraging results and suggests that we are on the right track when developing our system.

What we need to investigate further is how verification of the rule base can be improved, i.e. checking the rule set for soundness and completeness. Completeness is probably easy to check, if each vehicle get a maintenance plan from the rule set, for each of its components that should be maintained, the rule set is complete. Soundness is a bit harder as it involves some measurement of quality. In this case we are considering automatic detection of outliers, to point users towards potential errors in the rule set.

Somewhat related to automatic verification of the rule set is the use of Machine Learning (ML) (Mitchell, 1997) techniques for learning rules and supporting the users creating rules. In such a setting components (maintenance points) could be coupled with operational data and presented for the maintenance experts, their task would then be to label each components maintenance point with the three basic types of intervals.

After sufficiently many components have been given intervals we could then use ML to generalize from the examples to generate new rules for the rule set. This rule set could be expressed in the same format we described earlier, making the rule set a white box from a maintenance expert's perspective.

ACKNOWLEDGEMENT

This work has been funded by Scania CV AB and the Vinnova program for Strategic Vehicle Research and Innovation (FFI).

REFERENCES

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project.* Boston: Addison-Wesley.

Dunsdon, J., & Harrington, M. (2008). The Application of Open System Architecture for Condition Based Maintenance to Complete IVHM. *Aerospace Conference*, (pp. 1-9).

Durking, J. (1990). Application of Expert Systems in the Sciences. *Ohio Journal of Science*, *90*(5), 171-179.

Ian K. Jennions et al. (2011). *Integrated Vehicle Health Management: Perspectives on an Emerging Field.* Warrendale: SAE.

Kowalski, R. A., & Sadri, F. (2009). Integrating Logic Programming and Production Systems in Abductive Logic Programming Agents. In *Web Reasoning and Rule Systems* (pp. 1-23). Berlin: Springer.

Marsh, C. A. (1988). The ISA expert system: a prototype system for failure diagnosis on the space station. *Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems. 1*, pp. 60-74. Tullahoma: ACM.

Mats Carlsson et al. (2013). *SICStus Prolog Users's Manual.* Kista: Intelligent Systems Laboratory, Swedish Institute of Computer Science.

Mitchell, T. (1997). *Machine Learning.* New York: McGraw-Hill.

Red Hat. (2013, 09 25). *Drools*. (Red Hat) Retrieved 09 25, 2013, from http://labs.jboss.com/drools

Russel, S. J., & Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3rd edition).* Upper Saddle River: Pearson Education.

Senlin Liang, et al. (2009). OpenRuleBench: an analysis of the performance of rule engines. *Proceedings of the 18th international conference on World Wide Web.* Madrid.

Shapiro, S. C. (1987). Processing, bottom-up and top-down. In *Encyclopedia of Artificial Intelligence* (pp. 779-785). New York: John Wiley & Sons.

W3C. (2013, 02 05). *RIF RULE INTERCHANGE FORMAT CURRENT STATUS*. (W3C) Retrieved 08 21, 2013, from http://www.w3.org/standards/techs/rif#w3c_all